ADMINISTRATIVE MINUTIAE

- HW2 Due tonight
- HW3 out this afternoon
- OQ4 Out
- Midterm
  - Fill out piazza quiz before tomorrow
DATALOG: FACTS AND RULES

Facts = tuples in the database

Rules = queries

Actor(id, fname, lname)
Casts(pid, mid)
Movie(id, name, year)

Schema
DATALOG: FACTS AND RULES

Facts = tuples in the database

Actor(344759, ‘Douglas’, ‘Fowley’).
Casts(344759, 29851).
Casts(355713, 29000).
Movie(29445, ‘Ave Maria’, 1940).

Rules = queries

Q1(y) :- Movie(x,y,z), z=’1940’.
Q2(f, l) :- Actor(z,f,l), Casts(z,x), Movie(x,y,’1940’).
Q3(f,l) :- Actor(z,f,l), Casts(z,x1), Movie(x1,y1,1910),
          Casts(z,x2), Movie(x2,y2,1940).

Extensional Database Predicates = EDB = Actor, Casts, Movie
Intensional Database Predicates = IDB = Q1, Q2, Q3
Here are *unsafe* datalog rules. What’s “unsafe” about them?

\[\text{U1}(x,y) : \text{ParentChild}(“Alice”,x), y \neq “Bob”\]

\[\text{U2}(x) : \text{ParentChild}(“Alice”,x), \lnot \text{ParentChild}(x,y)\]
Here are unsafe datalog rules. What’s “unsafe” about them?

U1(x,y) :- ParentChild("Alice",x), y ≠ "Bob"

U2(x) :- ParentChild("Alice",x), !ParentChild(x,y)

Holds for every y other than "Bob"

U1 = infinite!
Here are *unsafe* datalog rules. What’s “unsafe” about them?

\[
U1(x,y) : \text{ParentChild}("Alice",x), y \neq "Bob"
\]

\[
U2(x) : \text{ParentChild}("Alice",x), \neg \text{ParentChild}(x,y)
\]

Holds for every \(y\) other than “Bob”
\[U1 = \text{infinite!}\]

Want Alice’s childless children, but we get all children \(x\) (because there exists some \(y\) that \(x\) is not parent of \(y\))
Here are *unsafe* datalog rules. What’s “unsafe” about them?

- **U1(x,y)**: 
  \[ \text{ParentChild}(“Alice”,x), \ y \neq “Bob” \]

- **U2(x)**: 
  \[ \text{ParentChild}(“Alice”,x), \ \neg \text{ParentChild}(x,y) \]

A datalog rule is *safe* if every variable appears in some positive relational atom.
Datalog: Relational Database

- Datalog can express things RA cannot
  - Recursive Queries
- Can Datalog express all queries in RA?
RELATIONAL ALGEBRA OPERATORS

Union $\cup$, difference $-$

Selection $\sigma$

Projection $\pi$

Cartesian product $\times$, join $\bowtie$
OPERATORS IN DATALOG

• Suppose we want Q1(...) to contain all the values from F1(...) and F2(...)
OPERATORS IN DATALOG

- Suppose we want $Q_1(...)$ to contain all the values from $F_1(...)$ and $F_2(...)$
  - $Q_1(...) :- F_1(...)$
  - $Q_1(...) :- F_2(...)$
- What about for difference?
OPERATORS IN DATALOG

• Suppose we want $Q_1(\ldots)$ to contain all the values from $F_1(\ldots)$ and $F_2(\ldots)$
  • $Q_1(\ldots) :- F_1(\ldots)$
  • $Q_1(\ldots) :- F_2(\ldots)$

• What about for difference?
  • $Q_1(\ldots) :- F_1(\ldots), \neg F_2(\ldots)$
OPERATORS IN DATALOG

• Suppose we want Q1(...) to contain all the values from F1(...) and F2(...)  
  • Q1(...) :- F1(...)  
  • Q1(...) :- F2(...)  

• What about for difference?  
  • Q1(...) :- F1(...), !F2(...)  
  • The variables (...) in F1 and F2 must be the same, or else we have an unsafe rule.
OPERATORS IN DATALOG

• Projection, from the variables $R_1, R_2, \ldots, R_k$
  select some subset of the variables
OPERATORS IN DATALOG

• Projection, from the variables $R_1, R_2, \ldots, R_k$
  select some subset of the variables
  • $Q_1$(subset) :- Original(all_attributes)

• Selection: only return certain records from our knowledge base
OPERATORS IN DATALOG

• Projection, from the variables $R_1, R_2, ..., R_k$
  select some subset of the variables
  • $Q_1$(subset) :- Original(all_attributes)

• Selection: only return certain records from our knowledge base
  • $Q_1(\ldots)$ :- Original(\ldots), selection_criteria
OPERATORS IN DATALOG

• Cross product: find all the pairs between $R(a_1,a_2\ldots)$ and $S(b_1,b_2\ldots)$
OPERATORS IN DATALOG

• Cross product: find all the pairs between $R(a_1,a_2...)$ and $S(b_1,b_2...)$
  • $Q1(a_1,b_1,a_2,b_2...) :- R(a_1,a_2...), S(b_1,b_2...)$
• Joins?
  • Natural
OPERATORS IN DATALOG

• Cross product: find all the pairs between $R(a_1,a_2\ldots)$ and $S(b_1,b_2\ldots)$
  • $Q1(a_1,b_1,a_2,b_2\ldots) :- R(a_1,a_2\ldots), S(b_1,b_2\ldots)$
• Joins?
  • Natural: $Q1(a,b,c) :- R(a,b), S(b,c)$
  • Theta
OPERATORS IN DATALOG

• Cross product: find all the pairs between \( R(a_1,a_2...) \) and \( S(b_1,b_2...) \)
  • \( Q1(a_1,b_1,a_2,b_2...) :- R(a_1,a_2...), S(b_1,b_2...) \)
• Joins?
  • Natural: \( Q1(a,b,c) :- R(a,b), S(b,c) \)
  • Theta: Cross product with selection
  • Equijoin: subset of Theta join
(SELECT Q.sno FROM Supplier Q WHERE Q.sstate = 'WA')
EXCEPT
(SELECT P.sno FROM Supply P WHERE P.price > 100)

EXAMPLE

\[ \text{Supply} \sigma_{\text{sstate} = \text{WA}} \text{Supplier} \sigma_{\text{Price} > 100} \]

\[ \pi_{\text{sno}} \text{Supplier} \]

\[ \pi_{\text{sno}} \text{Supply} \]
EXAMPLE

Datalog:

\[ \sigma_{\text{sstate} = 'WA'} \sigma_{\text{Price} > 100} \]

\[
\pi_{\text{sno}} \pi_{\text{sno}} \\
\text{Supplier}(\text{sno}, \text{name}, \text{city}, \text{sstate}) \\
\text{Part}(\text{pno}, \text{pname}, \text{psize}, \text{pcolor}) \\
\text{Supply}(\text{sno}, \text{pno}, \text{price})
\]
Datalog:

Q1(no, name, city, state) :-
  Supplier(sno, sname, scity, sstate),
  sstate='WA'
Q2(no, pno, price) :-
  Supply(s, pn, pr),
  pr > 100
Q3(sno) :- Q1(sno, n, c, s)
Q4(sno) :- Q2(sno, pn, pr)
Result(sno) :- Q3(sno),
  !Q4(sno)
MORE EXAMPLES W/O RECURSION

Find Joe's friends, and Joe's friends of friends.

\[
A(x) :- \text{Friend('Joe', x)} \\
A(x) :- \text{Friend('Joe', z), Friend(z, x)}
\]
MORE EXAMPLES W/O RECURSION

Find all of Joe's friends who do not have any friends except for Joe:

JoeFriends(x) :- Friend('Joe',x)
NonAns(x) :- JoeFriends(x), Friend(x,y), y != 'Joe'
A(x) :- JoeFriends(x), NOT NonAns(x)
MORE EXAMPLES W/O RECURSION

Find all people such that all their enemies' enemies are their friends

Q: if someone doesn't have any enemies nor friends, do we want them in the answer?
A: Yes!

`Everyone(x) :- Friend(x,y)`
`Everyone(x) :- Friend(y,x)`
`Everyone(x) :- Enemy(x,y)`
`Everyone(x) :- Enemy(y,x)`
`NonAns(x) :- Enemy(x,y), Enemy(y,z), NOT Friend(x,z)`
`A(x) :- Everyone(x), NOT NonAns(x)`
MORE EXAMPLES W/O RECURSION

Find all persons x that have a friend all of whose enemies are x's enemies.

\[ \text{Everyone}(x) : \neg \text{Friend}(x,y) \]
\[ \text{NonAns}(x) : \neg \text{Friend}(x,y) \text{ Enemy}(y,z), \neg \text{Enemy}(x,z) \]
\[ A(x) : \text{Everyone}(x), \neg \text{NonAns}(x) \]
MORE EXAMPLES W/ RECURSION

Two people are in the same generation if they are siblings, or if they have parents in the same generation.

Find all persons in the same generation with Alice.
MORE EXAMPLES W/ RECURSION

Find all persons in the same generation with Alice
Let’s compute $\text{SG}(x,y) = \text{"}x,y \text{ are in the same generation}\text{"}$

\[
\begin{align*}
\text{SG}(x,y) &\text{ :- ParentChild}(p,x), \text{ParentChild}(p,y) \\
\text{SG}(x,y) &\text{ :- ParentChild}(p,x), \text{ParentChild}(q,y), \text{SG}(p,q) \\
\text{Answer}(x) &\text{ :- SG("Alice", x)}
\end{align*}
\]
DATALOG SUMMARY

EDB (base relations) and IDB (derived relations)

Datalog program = set of rules

Datalog is recursive

Some reminders about semantics:

• Multiple atoms in a rule mean join (or intersection)
• Variables with the same name are join variables
• Multiple rules with same head mean union
CLASS OVERVIEW

Unit 1: Intro
Unit 2: Relational Data Models and Query Languages
Unit 3: Non-relational data
  • NoSQL
  • Json
  • SQL++
Unit 4: RDMBS internals and query optimization
Unit 5: Parallel query processing
Unit 6: DBMS usability, conceptual design
Unit 7: Transactions
Unit 8: Advanced topics (time permitting)